UECM1703 Introduction to Scientific Computing Marking Guide

TOPIC 5 Practical

UNIVERSITI TUNKU ABDUL RAHMAN

Faculty:	FES	Unit Code:	UECM1703
Course:	AM &FM	Unit Title:	Introduction Tt Scientific Computing
Year:	1&2	Lecturer:	Dr Yong Chin Khian
Session:	Oct 2022		

Q1. Consider the following linear system:

$$\begin{aligned} 32.1w + 20.9x + 21.4y + 18.6z &= 98.8 \\ 28.8w + 32.4x + 11.6y + 49.4z &= 83.4 \\ 69.8w + 49.5x + 44.4y + 15.0z &= 138.6 \\ 74.0w + 83.0x + 53.1y + 86.9z &= 85.0 \end{aligned}$$

(a) Write the above system in the form $\mathbf{AX} = \mathbf{b}$.

Ans.

$$\begin{bmatrix} 32.1 & 20.9 & 21.4 & 18.6 \\ 28.8 & 32.4 & 11.6 & 49.4 \\ 69.8 & 49.5 & 44.4 & 15.0 \\ 74.0 & 83.0 & 53.1 & 86.9 \end{bmatrix} \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 98.8 \\ 83.4 \\ 138.6 \\ 85.0 \end{bmatrix}$$

(b) Obtain the solution to the system above using matrix inversion.

Ans.

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

$$= \begin{bmatrix} 0.0326 & 0.0594 & 0.0221 & -0.0446 \\ -0.1452 & 0.0002 & 0.0414 & 0.0238 \\ 0.0923 & -0.0964 & -0.0487 & 0.0434 \\ 0.0546 & 0.0081 & -0.0286 & 0.0002 \end{bmatrix} \begin{bmatrix} 98.8 \\ 83.4 \\ 138.6 \\ 85.0 \end{bmatrix}$$

$$= \begin{bmatrix} 7.4456 \\ -6.562 \\ -1.9829 \\ 2.1169 \end{bmatrix}$$

```
import numpy as np
A = np.array([[32.1,20.9,21.4,18.6],[28.8,32.4,11.6,49.4],
[69.8,49.5,44.4,15.0],[74.0,83.0,53.1,86.9]])
b = np.array([[98.8],[83.4],[138.6],[85.0]])
x = np.linalg.inv(A)@b
print(x)
Output:
```

```
[[7.4456]
 [-6.562]
 [-1.9829]
 [2.1169]]
```

(c) Compute $95.8w + 116.6x + 106.4y + 84.0z$.

Ans.

$95.8w + 116.6x + 106.4y + 84.0z = 95.8(7.4456) + 116.6(-6.562) + 106.4(-1.9829) + 84.0(2.1169) = -85.0017$

```
import numpy as np
A = np.array([[32.1,20.9,21.4,18.6],[28.8,32.4,11.6,49.4],
[69.8,49.5,44.4,15.0],[74.0,83.0,53.1,86.9]])
b = np.array([[98.8],[83.4],[138.6],[85.0]])
x = np.linalg.inv(A)@b
anew = np.array([[95.8,116.6,106.4,84.0]])
xnew = np.inner(x, anew)
print("xnew = ", xnew)
Output:
xnew = -85.0017
```

Q2. You are given the following data:

No.	x_1	x_2	x_3
1	16.2	1.5	72.8
2	18.5	1.8	84.5
3	10.7	0.9	45.6
4	19.5	1.9	89.5
5	13.7	1.3	60.6
6	7.7	0.6	30.3
7	12.5	1.1	54.7
8	19.0	1.8	87.1
9	17.6	1.7	80.1
10	9.9	0.8	41.7

(a) Derive the sample covariance matrix($\mathbf{S_n}$) using the NumPy package.

Ans.

$\mathbf{S_n}$ = [[1.78023333e+01 1.97755556e+00 8.90103333e+01] [1.97755556e+00 2.20444444e-01 9.88711111e+00] [8.90103333e+01

9.88711111e+00 4.45065444e+02]]

```
import numpy as np
x1 = [16.2,18.5,10.7,19.5,13.7,7.7,12.5,19.0,17.6,9.9]
x2 = [1.5,1.8,0.9,1.9,1.3,0.6,1.1,1.8,1.7,0.8]
x3 = [72.8,84.5,45.6,89.5,60.6,30.3,54.7,87.1,80.1,41.7]
data = np.array([x1, x2, x3])
cov_matrix = np.cov(data, bias=False)
print("Cov(S_n)=",cov_matrix)
Output:
Cov(S_n)= [[1.78023333e+01 1.97755556e+00 8.90103333e+01]
 [1.97755556e+00 2.20444444e-01 9.88711111e+00]
 [8.90103333e+01 9.88711111e+00 4.45065444e+02]]
```

(b) Find the eigen values and eigen vectors of $\mathbf{S_n}$.

Ans.

Eigen values of $\mathbf{S_n}$ = [4.63086635e+02 6.06037060e-04 9.80911064e-04]
Eigen vectors of $\mathbf{S_n}$ = [[0.19606413 -0.70946401 0.67691926] [0.02177854 0.69329782 0.72032204] [0.9803492 0.126487 -0.15138192]]

```
import numpy as np
x1 = [16.2,18.5,10.7,19.5,13.7,7.7,12.5,19.0,17.6,9.9]
x2 = [1.5,1.8,0.9,1.9,1.3,0.6,1.1,1.8,1.7,0.8]
x3 = [72.8,84.5,45.6,89.5,60.6,30.3,54.7,87.1,80.1,41.7]
data = np.array([x1, x2, x3])
cov_matrix = np.cov(data, bias=False)
w,v = np.linalg.eig(cov_matrix)
print("Eigen values of S_n =",w)
print("Eigen vector of S_n =",v)

Output:
Eigen values of S_n = [4.63086635e+02 6.06037060e-04 9.80911064e-04]
Eigen vector of S_n = [[ 0.19606413 -0.70946401  0.67691926]
 [ 0.02177854  0.69329782  0.72032204]
 [ 0.9803492   0.126487   -0.15138192]]
```

(c) Find the largest eigen value.

Ans.

Largest eigen value = 463.0866352740975

```
import numpy as np
x1 = [16.2,18.5,10.7,19.5,13.7,7.7,12.5,19.0,17.6,9.9]
x2 = [1.5,1.8,0.9,1.9,1.3,0.6,1.1,1.8,1.7,0.8]
x3 = [72.8,84.5,45.6,89.5,60.6,30.3,54.7,87.1,80.1,41.7]
data = np.array([x1, x2, x3])
cov_matrix = np.cov(data, bias=False)
w,v = np.linalg.eig(cov_matrix)
LV = np.max(w)
print("Largest eigen value = ",LV)
Output:
Largest eigen value =  463.0866352740975
```

Q3. A researcher in a scientific foundation wished to evaluate the relation between intermediate and senior level annual salaries of bachelor's and master's level mathematician (Y, in thousand dollars) and index of work quality (X_1), number of years of experience (X_2), and index of publication success (X_3). Assume that regression model $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$ with independent normal error terms is appropriate.

y	x_1	x_2	x_3	y	x_1	x_2	x_3
33.2	3.5	9.0	6.1	40.3	5.3	20.0	6.4
38.7	5.1	18.0	7.4	46.8	5.8	33.0	6.7
41.4	4.2	31.0	7.5	37.5	6.0	13.0	5.9
39.0	6.8	25.0	6.0	40.7	5.5	30.0	4.0
30.1	3.1	5.0	5.8	52.9	7.2	47.0	8.3
38.2	4.5	25.0	5.0	31.8	4.9	11.0	6.4
43.3	8.0	23.0	7.6	44.1	6.5	35.0	7.0
42.8	6.6	39.0	5.0	33.6	3.7	21.0	4.4
34.2	6.2	7.0	5.5	48.0	7.0	40.0	7.0
38.0	4.0	35.0	6.0	35.9	4.5	23.0	3.5
40.4	5.9	33.0	4.9	36.8	5.6	27.0	4.3
45.2	4.8	34.0	8.0	35.1	3.9	15.0	5.0

You are given that:

- n is the number of observations.
- p is the number of parameters in the model.
- $\widehat{\hat{\boldsymbol{\beta}}} = \begin{bmatrix} \hat{\beta}_0 & \hat{\beta}_1 & \hat{\beta}_2 & \hat{\beta}_3 \end{bmatrix}$
- $SSR = \widehat{\boldsymbol{\beta}}^T \mathbf{X^T y} - \frac{1}{n}\mathbf{y^T J y}$, where $\mathbf{J}$ is an $n \times n$ matrix of one.

- $SSE = \mathbf{y^T y} - \widehat{\boldsymbol{\beta}}^T \mathbf{X^T y}$.
- $SST = \mathbf{y^T y} - \frac{1}{n}\mathbf{y^T J y}$.
- $MSE = \frac{SSE}{n-p}$
- $SE(\hat{\beta}_j) = \sqrt{MSE \times C_{jj}}$, where C_{jj} is the diagonal element of the $(\mathbf{X^T X})^{-1}$ corresponding to $\hat{\beta}_j$.

(a) Write the Python commands and output using matrix formulation to obtain the estimate of $\boldsymbol{\beta} = \begin{bmatrix}\beta_0 & \beta_1 & \beta_2 & \beta_3\end{bmatrix}$.

Ans.

$\widehat{\boldsymbol{\beta}} = \begin{bmatrix}\hat{\beta}_0 & \hat{\beta}_1 & \hat{\beta}_2 & \hat{\beta}_3\end{bmatrix} = [17.84693063648922\ 1.1031303951396438\ 0.32151968144970144\ 1.2$

```
import numpy as np
y = np.array([33.2, 40.3, 38.7, 46.8, 41.4, 37.5, 39, 40.7, 30.1, 52.9,
              31.8, 43.3,44.1, 42.8, 33.6, 34.2, 48,38,35.9,40.4,36.8,45
n = len(y)
x0 = np.array(np.repeat(1,n)).reshape(-1,1)
x1 = np.array([3.5,5.3,5.1,5.8,4.2,6,6.8,5.5,3.1,7.2,4.5,4.9,8,
               6.5,6.6,3.7,6.2,7,4,4.5,5.9,5.6,4.8,3.9]).reshape(-1,1)
x2 = np.array([9,20,18,33,31,13,25,30,5,47,25,11,23,35,39,21,7,
               40,35,23,33,27,34,15]).reshape(-1,1)
x3 = np.array([6.1,6.4,7.4,6.7,7.5,5.9,6,4,5.8,8.3,5,6.4,7.6,7,5,4.4,
               5.5,7,6,3.5,4.9,4.3,8,5]).reshape(-1,1)
x = np.hstack((x0, x1,x2,x3))
p = 4
xtx = x.T@x
xty = x.T@y
ixtx = np.linalg.inv(xtx)
b = ixtx@xty
print("beta hat =", b)
Output:
beta hat = [17.84693063648922 1.1031303951396438 0.32151968144970144 1.2
```

(b) Determine the predicted value for the mean score of Y with $X_1 = [5.3]$, $X_2 = [20]$, and $X_3 = [6.4]$.

Ans.

$\hat{Y} = 17.8469 + 1.1031([5.3]) + 0.3215([20]) + 1.2889([6.4]) = 38.3731$

```
xh = np.array([[1],[5.3], [20], [6.4]])
yh = xh.T@b
print("y hat=", yh)
Output:
```

```
y hat= 38.3731
```

(c) Write the Python commands and outputs to calculated SSR.

Ans. $SSR = 627.8169963651671$

```
bxty = b.T@xty
J = np.ones((n,n))
ytJy = y.T@J@y
SSR = bxty-ytJy/n
print("SSR=", SSR)

Output:
SSR= 627.8169963651671
```

(d) Write the Python commands and outputs to calculated SSE.

Ans.

```
yty = y.T@y
bxty = b.T@xty
SSE = yty-bxty
print("SSE=", SSE)

Output:
SSE= 61.44300363482762
```

(e) Write the Python commands and outputs to calculated SST.

Ans.

```
yty = y.T@y
J = np.ones((n,n))
ytJy = y.T@J@y
SST = yty-ytJy/n
print("SST=",SST)

Output:
SST= 689.2599999999948
```

(f) You are given that $R^2 = \frac{SSR}{SST}$ and adjusted R^2, $R^2_{Adj} = 1 - \frac{SSE/(n-p)}{SST/(n-1)}$. Write the Python commands and outputs to calculated R^2 and adjusted R^2.

Ans.

```
import numpy as np
yty = y.T@y
bxty = b.T@xty
J = np.ones((n,n))
ytJy = y.T@J@y
SSR = bxty-ytJy/n
SSE = yty-bxty
SST = yty-ytJy/n
print("SSE=", SSE)
print("SSR=", SSR)
print("SST=",SST)
RSq = SSR/SST
AdjRSq = 1-(SSE/(n-p))/(SST/(n-1))
print("Rsq=",RSq)
print("AdjRsq=", AdjRSq)

Output:
SSE= 61.44300363482762
SSR= 627.8169963651671
SST= 689.2599999999948
RSq= 0.9108565655415546
AdjRSq= 0.8974850503727878
```

(g) Suppose you are interested to test whether the number of years of experience (X_2) affect salaries, your hypotheses are $H_0 : \beta_2 = 0$ versus $H_1 : \beta_2 > 0$. The corresponding test statistic for testing these hypotheses is $t = \frac{\hat{\beta}_2}{SE(\hat{\beta}_2)}$. Write the Python commands and outputs to calculated t.

Ans.

```
import numpy as np
from numpy import sqrt,linalg
n = len(y)
p = 4
MSE = SSE/(n-p)
Cjj = ixtx[2,2]
SEbeta = sqrt(MSE*Cjj)
print("SEbeta = ", SEbeta)
t = b[2]/SEbeta
print("t = ",t)
Output:
```

```
SEbeta = 0.03710864908506008
t = 8.664278796911125
```

(h) Write the Python commands and outputs to calculated the p-value of the test in part(g).

Ans.

```
from scipy import stats
T = stats.t(n-p)
pvalue = 1-T.cdf(t)
print("p-value = ", pvalue)
Output:
p-value = 1.6634593036357614e-08
```

(i) Write the Python commands and outputs to calculated the test statistic for testing the hypothetses $H_0 : \beta_1 = \beta_2 = \beta = 3 = 0$ versus H_1 : At least one of the β's $\neq 0$.

Ans.

```
MSR = SSR/p
F = MSR/MSE
print("F = ", F)
Output:
F = 51.08938033827686
```

(j) Write the Python commands and outputs to calculated the p-value of the test in part(i).

Ans.

```
FD = stats.f(p,n-p)
pv2 = 1-FD.cdf(F)
print("p-value = ", pv2)
Output:
p-value = 3.203193266188009e-10
```

Q4. In an experiment to investigate the effect of color paper (yellow, green and red) on response rates for questionnaires distributed by the ”windshield method” in supermarket parking lots, 12 representative supermarket parking lots were chosen in a metropolitan area and each color was assigned random to four of the lots. The reponse rates (in percent) follow.

		j			
i		1	2	3	4
1	Yellow	28	26	31	33
2	Green	24	29	25	26
3	Red	33	38	36	39

Consider the model $y_{ij} = \mu_i + \epsilon_{ij}, i = 1, 2, 3; j = 1, 2, 3, 4$, where

- y_{ij} is the observed response rate for the i^{th} color paper assigned to the j^{th} parking lot.
- μ_i is the mean response rates of i^{th} color paper.
- $\epsilon_{ij} \sim N(0, \sigma^2)$

Use this model to answer the following questions.

(a) Let $\boldsymbol{\beta} = (\mu_1, \mu_2, \mu_3)^T$, $\mathbf{y} = [y_{11}, y_{12}, y_{13}, y_{14}, y_{21}, y_{22}, y_{23}, y_{24}, y_{31}, y_{32}, y_{33}, y_{34}]^T$, and $\boldsymbol{\epsilon} = [\epsilon_{11}, \epsilon_{12}, \epsilon_{13}, \epsilon_{14}, \epsilon_{21}, \epsilon_{22}, \epsilon_{23}, \epsilon_{24}, \epsilon_{31}, \epsilon_{32}, \epsilon_{33}, \epsilon_{34}]^T$. The above model can be express in the form $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$. Write down the matrix of $\mathbf{X}$ in kronecker form.

Ans.

$\mathbf{X} = \mathbf{I_3} \otimes \mathbf{1_4}$

(b) Write down the Python codes and the corresponding output to obtain the estimate of $\boldsymbol{\beta}$, $\mathbf{b}$.

Ans.

```
import numpy as np
y1 = [28,26,31,33]
y2 = [24,29,25,26]
y3 = [33,38,36,39]
y = np.hstack((y1,y2,y3))
I3 = np.eye(3)
One4 = np.ones(4).reshape(-1,1)
x = np.kron(I3,One4)
xtx = x.T@x
ixtx = np.linalg.inv(xtx)
xty = x.T@y
b = ixtx@xty
print("b=",b)
```

```
Output:
b= [29.5 26.  36.5]
```

(c) The hypotheses to test the equalities of means are $H_0 : \mu_1 = \mu_2 = \mu_3$ versus H_1: at least one population mean is different from the rest. H_0 can be express as $H_0 : \mathbf{C}\boldsymbol{\beta} = \mathbf{d}$. Determine a matrix $\mathbf{C}$ so that $H_0 : \mathbf{C}\boldsymbol{\beta} = \mathbf{0}$.

Ans.

$\mathbf{C} = [\mathbf{I_2}| - \mathbf{1_2}]$

(d) Continue using the Python codes from part (b), write down the Python codes and the corresponding output to compute the sum of squares of teaching method, SST.

Ans.

```
I2 = np.eye(2)
One2 = np.ones(2).reshape(-1,1)
C = np.hstack((I2,-One2))
print("C=",C)
Cb = C@b
print(Cb)
SST = Cb.T@np.linalg.inv((C@ixtx@C.T)) @Cb
print("SST = ",SST)

Output:
SST =  228.66666666666666
```

(e) Continue using the Python codes from part (b), write down the Python codes and the corresponding output to compute the sum of squares of error, SSE.

Ans.

```
yhat = x@b
e = y-yhat
#print("e =",e)
SSE = e.T@e
print("SSE=",SSE)

Output:
SSE= 64.0
```

(f) Continue using the Python codes from part (b), (d) and (e), write down the Python codes and the corresponding output to compute the test statistic.

Ans.

```
k = 3
N = 12
F = (SST/(k-1))/(SSE/(N-k))
print("F=",F)

Output:
F= 16.078125
```

(g) Write down the Python codes and the corresponding output to compute the p-value.

Ans.

```
from scipy import stats
FD = stats.f(k-1,N-k)
pvalue = 1-FD.cdf(F)
print("p-value=",pvalue)
```

Output: p-value= 0.0010693754449507822